IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application No.: 10/614,970	§	Examiner: Geib, Benjamin P.
Filed: July 8, 2003	§	Group/Art Unit: 2181
Inventor:	§	Atty. Dkt. No: 5500-81600/TT4899
Mitchell Alsup	§	
Gregory W. Smaus	§	
	§	
	§	
Title: System and Method of Implementing Microcode Operations as Subroutines	§	

REPLY BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

This brief is in reply to the Examiner's Answer dated March 29, 2007. Appellants respectfully request that this Reply Brief be entered pursuant to 37 C.F.R. § 41.41 and considered by the Board of Patent Appeals and Interferences.

REPLY

First ground of rejection:

Claims 1, 15-17, 27-29, 39 and 41 stand finally rejected under 35 U.S.C. § 102(b) as being anticipated by Tran (U.S. Patent 5,864,689) (also referencing "Intel Architecture Software Developer's Manual (hereinafter "Manual"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1, 15, 16, 17, 27, 28, and 39:

Regarding claim 1, contrary to the Examiner's assertion, Tran clearly fails to disclose *a scheduler coupled to the dispatch unit and configured to schedule dispatched operations for execution, wherein in response to receiving a microcoded instruction, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction*.

Appellants' arguments from the Appeal Brief filed September 25, 2006 regarding this rejection are herein incorporated by reference. As explained in Appellants' Appeal Brief, the x86 CALL instruction is not a microcoded instruction, as would be easily understood by anyone of ordinary skill in the art, and Tran does not teach that the CALL instruction itself is a microcoded instruction (i.e., one executed in microcode). Instead, the CALL instruction is described as an example of a branch instruction, which also cannot be considered a microcoded instruction (see, e.g., Tran claim 5, and column 4, lines 44 – 54, in which a standard subroutine call and return are described.) Tran teaches that if a branch type opcode (including a CALL opcode, for example) is detected and the target address of the branch is within a particular range of addresses, the target address may be routed to microcode unit 45. In other words, Tran does not describe detecting a **microcoded** instruction, but detecting a non-microcoded instruction that is used to

emulate an instruction that is not included in the instruction set (e.g., a DSP function for which no instruction exists). The concept of a microcoded instruction is well understood in the art. No one of ordinary skill in the art would consider the x86 CALL instruction to be described in Tran as a microcoded instruction. To the contrary, the x86 CALL instruction in Tran is clearly described as not being a microcoded instruction. Instead, it is the target of the CALL instruction that may be routed to the microcode unit in Tran.

As further explained in Appellants' Appeal Brief, Tran does not teach dispatching to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction. Tran teaches that if a non-microcoded instruction (e.g., the x86 CALL instruction) is detected that has a target address in a particular range of addresses, the target address (not the CALL instruction itself or another microcode subroutine call) is routed to the microcode unit, along with an "indication of the instruction", which may also include operands for the instruction. This is clearly not the same as a dispatch unit dispatching to the scheduler a microcode subroutine call operation in response to *receiving a microcoded instruction*. An indication of an instruction, even one that includes a target address of a subroutine call instruction, is not itself a microcode subroutine call operation, as recited in claim 1.

In the Response to Arguments section of the Final Office Action, the Examiner submitted:

> The execution of the x86 CALL instruction involves the storing of context information (such as the instruction pointer) in addition to the execution of the routine (microcode subroutine instructions in the context of Tran)... In order for context information to be stored, the x86 CALL instruction itself must be executed by one of the execution units of the processor. Execution by one of the execution units requires that the instruction be dispatched from decode unit (part of the dispatch unit) to the reservation station (i.e., the scheduler) associated with the desired execution unit.

Appellants disagree. There is nothing in Tran that describes that any of these operations (e.g., storing of context information) is performed when the CALL instruction is used to emulate an instruction that is not in the instruction set. In addition, the Examiner's remarks appear to acknowledge that the CALL instruction itself is not a

microcoded instruction for which the operations recited in claim 1 are performed. For example, the Examiner states that the "CALL instruction itself must be executed by one of the execution units," and that it is dispatched to the scheduler associated with the desired execution unit.

In addition, as explained in Appellants' Appeal Brief, if Tran taught dispatching the CALL instruction itself to the scheduler of an execution unit 38, as the Examiner suggests, this would be in direct contrast to the limitations of claim 1, in which *in response to receiving a microcoded instruction, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation*. This claim clearly indicates that the received microcoded instruction and the dispatched microcode subroutine call operation are not the same instruction/operation. In response to receiving one (the microcoded instruction), the other (the microcode subroutine call operation) is dispatched to the scheduler.

In his Answer, the Examiner disagrees with Appellants' arguments above, and submits that Appellants appear to be reading "microcoded instruction" too narrowly. Appellants respectfully disagree and assert that "microcoded instruction" is a term of art well understood by those of ordinary skill in the art. **Appellants submit that the Examiner is improperly attempting to re-define this term in order to support his rejection.** In his Answer, the Examiner submits that Tran gives an x86 CALL instruction as an example of an instruction to be performed by the microcode unit, and that thus, it is a microcoded instruction that is performed (a least in part) by the microcode unit (citing column 4, lines 52-59.) The Examiner's assertion is simply incorrect. This passage in Tran does not characterize the x86 CALL instruction itself as a microcoded instruction, but describes it as an exemplary subroutine call instruction, which Tran also describes as a branch-type instruction. No one of ordinary skill in the art would consider a branch-type instruction to be a microcoded instruction, as these instructions are executed directly by one of the execution units. They are not the type of complex operations for which a microcode routine is stored in microcode. **Tran itself teaches this in column 6, lines**

38-40, "For example, execute units 38 may include a branch execute unit for executing branch instructions."

The Examiner's citation in column 4 goes on to describe that, as discussed above, when this non-microcoded instruction has a target address in a particular range, the target address is routed to the microcode unit, where it identifies the microcode routine to be executed. Appellants assert that, contrary to the Examiner's suggestion, the CALL instruction itself is not executed by the execution units (or dispatched to the schedulers) in such cases (i.e., in cases in which the target address in within a particular range and the CALL instruction is used to emulate a DSP function.) This is clear in both the specification (as quoted above) and in the claims of Tran. For example, claim 1 of Tran recites, in part,

> wherein said instruction decode unit is further configured to detect said branch opcode of a given branch instruction and, in response to detecting said branch opcode, to either invoke said first microcode DSP function **or** provide said given branch instruction to said first execute unit which, in response, initiates a subroutine stored in a memory, depending upon said target address value of said given branch instruction (emphasis added).

In Tran, it is the operations of the microcode routine (not a microcode subroutine call itself) that is dispatched to the execution units to perform a DSP function. For example, see column 4, lines 15-30, which states, in part:

> Microcode unit 45 fetches instructions that correspond to a routine which performs the indicated DSP function from the read-only memory…. Instructions fetched by microcode unit 45 are conveyed to execute units 38 and load/store unit 40.

The Examiner submits that when decoding is applied to the x86 CALL instruction the instruction is transformed into a second format that the Examiner refers to as the claimed "microcode subroutine call operation." **The Examiner's statement is pure speculation completely unsupported in the cited art. There is no description of any such microcode subroutine call operation resulting from the decoding of the CALL instruction.** Appellants assert that it is clear from the specification and claims of Tran that the flow previously described by the Examiner for decoding and execution of a

CALL instruction in his Answer (including his remarks referencing the Manual) applies only when the CALL instruction is actually being used as a standard subroutine CALL instruction, not when it is used to invoke execution of a DSP function stored in microcode. Tran teaches instead that when a CALL instruction with a target address in a particular range is detected, operations for executing a DSP function are fetched beginning at the target address. This would not require a microcode subroutine call in the system of Tran, as the instruction decoder includes circuitry for routing the target address to the microcode unit (column 4, lines 54-59) so that the operations of the DSP function can be fetched and executed in sequence until reaching the end of the microcode sequence. At that point, a sequencer in the microcode unit returns a control signal that indicates that all of the operations in the DSP function have been dispatched (i.e., so that the instruction decoder may resume fetching and decoding instructions from memory). Nothing in this description of the operation of Tran describes anything about a microcode subroutine call, much less dispatching one to the schedule. Therefore, the use of a branch-type instruction to invoke a DSP function stored in microcode, as taught by Tran, clearly does not teach all the limitations of Appellants' claim 1.

Anticipation requires the presence in a single prior art reference disclosure of each and every limitation of the claimed invention, arranged as in the claim. M.P.E.P 2131; *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The **identical** invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). As discussed above, Tran clearly fails to disclose *in response to receiving a microcoded instruction, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction.* Therefore, Tran certainly cannot be said to anticipate claim 1.

For at least the reasons above, the rejection of claim 1 is not supported by the cited art and removal thereof is respectfully requested. Claim 17 includes limitations

similar to claim 1, and so the arguments presented above apply with equal force to this claim, as well.

Claim 29 and 41:

Regarding claim 29, as explained in Appellants' Appeal Brief, **Tran clearly fails to disclose *detecting a microcoded instruction within the stream of instructions, wherein the microcoded instruction immediately precedes an other instruction in program order, and in response to said detecting, dispatching a microcode subroutine call operation that identifies a microcode subroutine associated with the microcoded instruction.*** First, as discussed above, Tran does not teach detecting a microcoded instruction, but detecting a non-microcoded instruction that emulates a microcoded DSP function. The Examiner cites column 4, lines 36-41, as teaching detecting a microcoded instruction that immediately precedes another instruction in program order. However, this citation describes that if instruction decode unit 36 detects an instruction to be performed by microcode unit 45, it stalls until microcode unit 45 indicates that the routine corresponding to that instruction (not the instruction itself) has completed dispatch. This citation says nothing about the microcoded instruction immediately preceding another instruction in program order, as recited in Appellants' claim 29.

Further regarding claim 29, the Examiner cited column 8, line 55 – column 9, line 4, as teaching dispatching a microcode subroutine call operation in response to detecting the microcoded instruction. However, as described in Appellants' Appeal Brief and above, this citation does not describe dispatching a microcode subroutine call operation. Instead, this citation describes instruction decode unit 36 sending an indication of the detected instruction to microcode unit 45, which then completes the routine (i.e., the operations stored in microcode to implement a DSP function are dispatched by the microcode unit for execution). As discussed above regarding claim 1, sending an indication of a detected (non-microcoded) instruction is clearly not the same as dispatching a microcode subroutine call operation, as recited in Appellants' claim 29.

The Examiner previously cited column 4, lines 42-54 as teaching that the microcode subroutine call operation pushes an address of the other instruction onto a stack, that the microcode subroutine includes a return operation, and that execution of the return operation pops the address from the stack (x86 Call instruction and x86 Return instruction). However, this citation has nothing to do with pushing or popping an instruction onto or off of a stack as part of a microcode subroutine call operation dispatched in response to detecting a microcoded instruction, as recited in claim 29. Instead it describes that source-code level subroutine call instructions (e.g., the CALL instruction of the x86 instruction set) having target addresses within a particular range of addresses are indicative of DSP functions. The instruction detected by instruction decode unit 36 is not a microcoded instruction for which a microcode subroutine call operation is dispatched in response to this detection. It is an example of a branch-type instruction. Furthermore, as discussed above, the Examiner's descriptions of the execution flow of an x86 CALL instruction (including those involving instruction pointers) **do not apply** to the case when this instruction is used to invoked a microcode routine. Nothing in Tran discloses a microcode subroutine pushing and popping operations on a stack, as the microcoded DSP routines are not invoked using a microcode subroutine call.

The Examiner further submitted that the x86 CALL instruction is both a microcoded instruction and a microcode subroutine call operation. This is clearly illogical and is not taught by Tran. In Tran, the CALL instruction is not a microcode subroutine call operation, nor is it dispatched to a scheduler in response to detection of a microcoded instruction. Instead, Tran teaches that a target address of a CALL instruction (which is also clearly not a microcode subroutine call operation) is sent to microcode unit 45 in response to detection of a branch-type instruction (the CALL instruction itself). Appellants' claims clearly indicate that the received and/or detected microcoded instruction and the dispatched microcode subroutine call operation are not the same instruction/operation. In response to receiving and/or detecting one (the microcoded instruction), the other (the microcode subroutine call operation) is dispatched. **By contrast, in Tran, the x86 CALL instruction, a source code instruction of the x86 instruction set architecture, is received and is itself dispatched for execution only if**

the target address is not within a particular range. In the case in which it is used to invoke execution of a microcoded DSP function, it is not executed at all. In neither case is a microcode subroutine call operation dispatched, as discussed in detail in remarks regarding claim 1.

In his Answer, the Examiner disagrees with Appellants' arguments above, and submits that since the x86 CALL instruction saves an instruction pointer to the next instruction in program order, it precedes another instruction in program order. However, as discussed in detail above, the CALL instruction is not executed when used to invoke execution of a microcoded DSP function. Therefore, the actions described by the Examiner are not performed in this case. Appellants again assert that it is not inherent in Tran that a detected microcode instruction precedes another instruction in program order, as suggested by the Examiner.

In his Answer, the Examiner again submits that Tran has taught dispatching a microcode subroutine call operation of an x86 CALL instruction. Appellants disagree, for at least the reasons discussed above regarding claim 1.

Also in his Answer, the Examiner submits that Tran has taught executing an x86 CALL instruction, which involves the dispatch of a microcode subroutine call operation and that this microcode subroutine call operation (i.e., the "second format" of the x86 CALL instruction) involves saving the address of another instruction (i.e., the current instruction pointer.) Appellants again asset that this is completely unsupported in the cited art, as these operations are only performed for a standard CALL operation, not one used to invoke execution of a microcoded DSP function. The Examiner also submits that Tran has taught that a microcoded subroutine executed as part of an x86 CALL instruction concludes with the execution of a return instruction, RET, in column 4, lines 42-54. This is incorrect. This passage merely describes that the RET instruction (used to return from a standard subroutine call, not a microcoded DSP function) is also an example of a branch instruction. **This has absolutely nothing to do with the limitations of claim 29.**

For at least the reasons above, the rejection of claim 29 is not supported by the cited art and removal thereof is respectfully requested. Claim 41 includes limitations similar to claim 29, and so the arguments presented above apply with equal force to this claim, as well.

Second ground of rejection:

Claim 40 stands finally rejected under 35 U.S.C. § 102(b) as being anticipated by Carbine et al. (U.S. Patent 5,630,083) (hereinafter "Carbine"). Appellants traverse this rejection for at least the following reasons.

Carbine clearly fails to disclose *dispatching one or more operations included in a first microcode subroutine and one or more operations included in a second microcode subroutine, wherein said dispatching the one or more operations in the first microcode subroutine comprises performing register name replacements using replacement register names stored in a first alias table element and wherein said dispatching the one or more operations in the second microcode subroutine comprises performing register name replacements using replacement register names stored in a second alias table element.*

Appellants' arguments from the Appeal Brief filed September 25, 2006 regarding this rejection are herein incorporated by reference. The Examiner cited column 12, lines 35-56, as teaching generic microcode routines, and asserts that each of these generic microcode routines "has micro-alias registers" to replace register names within the routine. The Examiner seems to be implying that these generic routines, including what he interprets as separate micro-alias registers, teach dispatching operations from both a first and second microcode subroutine, and the use of both a first and second alias table element. However, there is nothing in this citation or elsewhere in Carbine that teaches multiple alias table elements or multiple micro-alias registers. Instead, Carbine describes a single micro-alias register (having multiple fields, but not multiple entries), which is

loaded with different data dependent on which of four CUOPs is selected by multiplexer 560. There is also nothing in this citation that teaches dispatching multiple microcode subroutines, as recited in claim 40. Instead, Carbine describes microcode sequencing unit 534 issuing multiple CUOPs for one microcode flow (associated with a single entry point) at a time (see, e.g., column 11, lines 11-13). Since Carbine fails to teach or suggest dispatching operations from a first and second microcode subroutine and a first and second alias table element, Carbine cannot be said to anticipate claim 40.

In the Response to Arguments section of the Final Office Action, the Examiner submits that Carbine specifically refers to a plurality of micro-alias registers at column 12, lines 36-39. The Examiner also submits that Carbine states that registers are not hard-coded into any routine, "thereby indicating multiple generic microcode routines are used. Therefore, Carbine has taught dispatching multiple generic microcode routines." First, Appellants assert that the Examiner has misinterpreted Carbine's reference to "micro-alias registers." The Examiner's citation at column 12, lines 36-39 states, "The micro-alias registers 562 are particularly useful in long instruction flows, which allows the microcode programmer flexibility in retaining information and simplifying code." As is apparent from reading the rest of this passage, Carbine is referring to the separate storage locations (i.e., the SRC1, SRC2, and DEST fields) within the micro-alias register, not to multiple sets of such register fields in a plurality of micro-alias registers. Carbine refers to "the micro-alias register 562" and "the micro-alias registers 562" interchangeably throughout the specification. **Carbine does not ever refer to multiple sets of such fields contained in multiple instances of micro-alias register 562, much less multiple instances of micro-alias register 562 each of which is associated with a different microcode routine, as the Examiner suggests.**

In addition, Appellants assert that even if multiple generic routines may exist in the system of Carbine, there is nothing in Carbine that teaches **dispatching two such routines.** While the system of Carbine is directed toward parallel decoding of multiple instructions (see Abstract) there is nothing in Carbine that discloses dispatching operations from two different microcode subroutines, as required in Claim 40.

Appellants assert that it is clearly not necessary that a system support dispatching operations from two different routines even if they may be decoded in parallel, and the Examiner has not cited anything in Carbine that discloses this limitation of Appellants' claim.

In his Answer, the Examiner disagrees with Appellants' arguments above, and again notes that column 12, line 36 includes the phrase "micro-alias registers 562." This has been addressed above. The Examiner also notes that Carbine describes multiple examples of generic microcode routines, and that the "registers are not hard coded into any routine." The Examiner suggests that because there may be a plurality of generic microcode routines stored in the microcode of Carbine, there must be a plurality of generic microcode subroutines being executed/dispatched (implying that each has its own set of micro-alias register fields). **However, this is completely unsupported in the cited art. Appellants again assert that nothing in Carbine describes dispatching operations from two different microcode routines and using a different alias table element (or set of micro-alias register fields) for each one.**

For at least the reasons above, the rejection of claim 40 is not supported by the cited art and removal thereof is respectfully requested.

Third ground of rejection:

Claims 2-6, 18-22 and 30-34 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran in view of Carbine. Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 2, 18, and 30:

Regarding claim 2, contrary to the Examiner's assertion, Tran in view of Carbine clearly fails to teach or suggest *wherein the dispatch unit is further configured to dispatch an operation that provides one or more register names for use as replacement register names within the microcode subroutine.* The Examiner admits that Tran does not explicitly teach this limitation and relies on Carbine to teach it. The Examiner submits that Carbine has taught a dispatch unit (microcode sequencing unit; FIG. 5, component 534) that is configured to dispatch an operation (LOADUAR signal) that provides one or more register names for use as replacement register names within the microcode subroutine (Carbine, column 12, lines 24-67).

Appellants' arguments from the Appeal Brief filed September 25, 2006 regarding this rejection are herein incorporated by reference. As explained in Appellants' Appeal Brief, the microcode sequencing unit of Carbine is clearly not analogous to the dispatch unit of Appellants' claims, as it does not perform the functions recited in claim 1. For example, the microcode sequencing unit is not *configured to dispatch operations*, much less to *dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction.*

In addition, Appellants assert that the LOADUAR ("Load Micro-Alias Register") signal is not an operation dispatched by a dispatch unit to a scheduler (i.e., an operation dispatched for execution by an execution unit). Instead, it is a signal supplied by the microcode sequencing unit 534 to load the micro-alias register from one of four Cuops (as selected by a UARUIP signal). Furthermore, this signal clearly does not provide register names for replacement register names within a microcode subroutine, and no such replacement is described in Carbine. Instead, Carbine describes that microcode subroutines may be written such that they may be shared by multiple microcode routines by coding them to be generic. In the example described in the Examiner's citation, a microcode routine stores the address of a register in which a number to be rounded is stored in the micro-alias registers and then a rounding subroutine is run, using the

contents of the micro-alias register to locate the number to be rounded. This has absolutely nothing to do with Appellants' claim, in which ***an operation*** *that provides one or more register names for use as replacement register names within the microcode subroutine* is dispatched by a dispatch unit to a scheduler for execution.

The Examiner submits that at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to modify the dispatch unit of Tran to dispatch an operation that provides one or more register names for use as replacement register names with the microcode subroutine as taught by Carbine and that the suggestion/motivation for doing so would have been that a generic microcode routine can be used by any number of other microcode programs (Carbine, column 12, lines 49-52). Appellants assert, however, that Carbine does not teach this limitation, as discussed above. Therefore, modifying Tran to include the LOADUAR signal, or allowing generic microcode routines to be shared, would not result in Appellants' claimed invention.

To establish a *prima facie* obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. Obviousness cannot be established by combining or modifying the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion or incentive to do so. *In re Bond,* 910 F. 2d 81, 834, 15 USPQ2d 1566, 1568 (Fed. Cir. 1990).

In his Answer, the Examiner disagrees with Appellants' arguments above, and submits that Appellants are reading, "dispatch unit" too narrowly. The Examiner submits that the microcode sequencing unit of Carbine sends (which he interprets as "dispatches") the LOADUAR signal that performs that operation of providing "one or more register names for use as replacement register names within the microcode subroutine" as claimed. The Examiner further submits that the LOADUAR signal loads a micro-alias register that holds an address of a register that is to be used within a microcode subroutine, and that therefore, the LOADUAR signal performs the operation as claimed and the microcode sequencing unit is analogous to the claimed dispatch unit. The

Examiner also submits that, in general, the execution of instructions in a microprocessor consists of nothing more than the sending/dispatching of various signals throughout the processor and that Appellants appear to be reading "operation" too narrowly. The Examiner states, "the LOADUAR signal of Carbine performs the claimed operation and is an operation." Appellants respectfully disagree.

Appellants assert that both "dispatch unit" and "operation" (in this context) are terms of art well understood by those of ordinary skill in the art, and that the phrase, "dispatch an operation," would also be well understood in the context of these terms of art. In Addition, the claims must be given their broadest reasonable interpretation that is consistent with Appellants' specification. Furthermore, the LOADUAR signal does not provide one or more register names, as the Examiner suggests. Instead, this signal merely loads (i.e., latches) the micro-alias register from one of several sources, where data is already stored. Appellants assert that no one of ordinary skill in the art having benefit of Appellants' specification would consider the LOADUAR signal to be the *operation that provides one or more register names for use as replacement register names within the microcode subroutine* (that is dispatched by a dispatch unit), that is recited in Appellants' claims.

For at least the reasons above, the rejection of claim 2 is not supported by the cited art and removal thereof is respectfully requested. Claims 18 and 30 include limitations similar to claim 2, and so the arguments presented above apply with equal force to these claims, as well.

Claims 3, 19, and 31:

Regarding claim 3, as explained in Appellants' Appeal Brief, Tran in view of Carbine clearly fails to teach or suggest *wherein the dispatch unit is configured to allocate an alias table element to store the one or more register names in response to handling the operation*. The Examiner cites Carbine (micro-alias register; FIG. 5, component 562; and column 12, lines 15-35) as teaching this limitation. The Examiner's

cited passage describes that the micro-alias register 562 is loaded with three values (in fields SRC1, SRC2, and DEST) from one or more Cuops, dependent on a 4:1 multiplexer 560. The micro-alias register is clearly not a table in which elements are allocated, much less one that is configured to allocate an element to store register names in response to handling an operation that provides one or more register names for use as replacement register names within the microcode subroutine and that is dispatched by a dispatch unit to a scheduler for execution. In addition, there is nothing in Carbine that teaches or suggests that a dispatch unit (i.e., one that performs the functions recited in Appellants' claims) allocates a micro-alias register for any reason. Instead, the micro-alias register is always available to be loaded with information from one of four Cuop registers 550. Therefore, Appellants assert that Tran and Carbine, taken alone or in combination, do not teach or suggest this limitation of claim 3.

In addition, the Examiner has failed to provide a reason why a person of ordinary skill in the art would be motivated to combine the teachings of Tran and Carbine in teaching the limitations of claim 3. The broad statements made by the Examiner regarding claim 2 do not include a suggestion in the cited prior art for such a specific combination and include no mention of the specific limitations of claim 3.

In his Answer, the Examiner disagrees with Appellants' arguments above, and submits that it is not the micro-alias register of Carbine that is indicated as being a table, but a micro-alias register is indicated as being an alias table element. The Examiner suggests that the plurality of micro-alias registers make up an alias table. **This statement is completely unsupported in the cited art**. As discussed above regarding claim 40, Carbine's use of the plural "micro-alias registers" refers to the separate storage locations (i.e., the SRC1, SRC2, and DEST fields) within a single micro-alias register, not to multiple sets of such register fields in a plurality of micro-alias registers. Carbine refers to "the micro-alias register 562" and "the micro-alias registers 562" interchangeably throughout the specification. **Contrary to the Examiner's repeated assertions, Carbine does not ever refer to multiple sets of such fields contained in multiple instances of micro-alias register 562.** Therefore, it appears that the Examiner is

attempting to equate the single instance of micro-alias register 562 with the alias table of Appellants' claims whose elements are individually allocated in response to handling the operation dispatched as in claim 2 (an operation that provides register names for replacement). Appellants assert that no one of ordinary skill in the art would consider a single register to be the alias table of Appellants' claims, nor does the single micro-alias register of Carbine teach the limitations of the alias table recited in claim 3. For example, elements in this register are not allocated to store register names provided by the operation recited in claim 2 (which Carbine also does not teach) in response to handling the operation. Instead, the values for the fields of the register are loaded (i.e., latched) from one of several sources in response to the LOADUAR signal being asserted. Contrary to the Examiner's assertion, sending a signal to load a register is clearly not the same as allocating a register, much less allocating a table element in an alias table as recited in claim 3, as would be understood by one of ordinary skill in the art.

For at least the reasons above, the rejection of claim 3 is not supported by the cited art and removal thereof is respectfully requested. Claims 19 and 31 include limitations similar to claim 3, and so the arguments presented above apply with equal force to these claims, as well.

Claims 4-5, 20-21, and 33-34:

As explained in Appellants' Appeal Brief, Tran in view of Carbine clearly fails to teach or suggest *wherein the dispatch unit is configured to maintain multiple allocated alias table elements at a same time* (as recited in claim 4) or *wherein each of the multiple allocated alias table elements is associated with a respective microcode subroutine, wherein the dispatch unit is configured to maintain each alias table element at least until all branch operations within the respective microcode subroutine have resolved* (as recited in claim 5.) The Examiner again cites the micro-alias register 562 of Carbine and column 12, lines 15-35 as teaching the alias table of these claims. However, as discussed above, the micro-alias register of Carbine is clearly not an element of an alias table, which is allocated in response to an operation that provides register names for

replacement register names within a microcode routine, as required by Appellants' claims. In addition, as discussed above regarding claim 40, Carbine does not teach or suggest multiple instances of micro-alias register 562, as was suggested by the Examiner. Therefore, Carbine clearly cannot, and does not, teach or suggest maintaining multiple allocated alias table elements at a same time, as recited in claim 4.

In addition, as explained in Appellants' Appeal Brief, Tran does not teach anything about each of multiple allocated alias table elements being associated with a respective microcode subroutine. The Examiner submits that Carbine teaches the dispatch unit is configured to maintain each alias table element at least until all branch operations within the respective microcode subroutine have resolved, "If a micro-branch (a branch in microcode) is mispredicted the dispatch unit updates/maintains the micro-alias registers so that execution can restart at the actual target of the micro-branch. Therefore, the dispatch unit maintains each micro-alias register until all branch operations within the respective microcode subroutine have resolved; See column 23, lines 12-35". **Appellants assert that the Examiner's citation, as well as his own remarks, teaches away from these limitations of Appellants' claims**. First, the Examiner's remarks include the phrase, "the dispatch unit updates/maintains the micro-alias registers." Appellants assert that updating the micro-alias registers is the **opposite** of maintaining alias table elements. The term "maintain" is used in this context to mean that the value is not changed, as is clear in both the recitation of Appellants' claims and in the accompanying description in Appellants' specification.

Similarly, the Examiner's citation in column 23 states, in part, that in response to a mispredicted branch, "If either macro-alias data or micro-alias data or both may be used by subsequent micro-operations, then the first instructions at the target microcode flow restore the state by restoring the macro-alias registers if necessary and by restoring the micro-alias registers if necessary for use by subsequent micro-operations." This clearly indicates that in Carbine, the micro-alias registers (i.e., the fields of micro-alias register 562) must be restored because the values were not maintained until the branch was

resolved. Therefore, Carbine teaches away from the above-referenced limitation of claim 5.

Appellants assert, therefore, that Tran and Carbine, taken alone or in combination, do not teach or suggest this limitation of claims 4 and 5.

In addition, the Examiner has failed to provide a reason why a person of ordinary skill in the art would be motivated to combine the teachings of Tran and Carbine in teaching the limitations of claims 4 and 5. The broad statements made by the Examiner regarding claim 2 do not include a suggestion in the cited prior art for such a specific combination and include no mention of the specific limitations of claims 4 and 5.

In his Answer, the Examiner disagrees with Appellants' arguments above, and again asserts that Carbine has taught maintaining a plurality of micro-alias registers and that each of these micro-alias registers is associated with the generic microcode subroutine for which it is holding a replacement register name. Appellants disagree. As discussed above and in remarks regarding claim 40, Carbine does not teach that multiple sets of micro-alias registers exist, much less that they are allocated at the same time for different generic subroutines. Instead the single micro-alias register can only hold register names for the different generic routines at different times. In fact, this is why, in the Examiner's citation above, Carbine describes that they must be restored after a mis-prediction (i.e., because they no longer contain the register names needed). **Clearly, this teaches that the register names are not maintained in the micro-alias register until all the branches are resolved or they would not need to be restored.**

The Examiner also submits that Appellants are reading the term, "maintain" too narrowly and that, "Clearly, updating is a type of maintenance." Appellants again assert that the claims must be interpreted in light of the specification and that the specification uses this term in reference to the alias table only when referring to the fact that each alias table element remains allocated to a given microcode subroutine (with no mention of any changes in value) until all branch operations in the corresponding microcode subroutine

have resolved. Appellants assert that the limitation that each one of the multiple allocated alias table elements is associated with a respective microcode subroutine and is maintained until all branches in that microcode subroutine have resolved is clearly not taught by Carbine, and that the Examiner's assertion that "updating is a type of maintaining" makes no functional sense in this context.

For at least the reasons above, the rejection of claims 4 and 5 is not supported by the cited art and removal thereof is respectfully requested. Claims 20-21 and 33-34 include limitations similar to those of claims 4 and 5, and so the arguments presented above apply with equal force to these claims, as well.

Claims 6, 22, and 32:

Regarding claim 6, as explained in Appellants' Appeal Brief, Tran in view of Carbine clearly fails to teach or suggest *wherein in response to detection of a branch misprediction within a microcode subroutine, the dispatch unit is configured to perform replacements within one or more microcode operations included within the microcode subroutine according to the one or more register names stored within a respective alias table element and to dispatch the one or more microcode operations subsequent to performing the replacements*. The Examiner again cites Carbine (column 23, lines 12-35) as teaching these limitations. However, as discussed above, Tran in view of Carbine does not teach or suggest the dispatch unit and alias table elements of Appellants' invention, or the register names stored in the alias table elements. In addition, this passage of Carbine describes that in order to restore the macro-alias and micro-alias registers after a branch misprediction, the macroinstruction must first be re-fetched and re-decoded (as illustrated in FIG. 12) and then the first instructions at the target microcode routine may restore the macro-alias and micro-alias registers, if they will be needed by subsequent micro-operations. This clearly **teaches away from** the limitations of claim 6, in which the one or more register names already stored within a respective alias table element are used to perform replacements in response to detection of a branch misprediction.

In his Answer, the Examiner disagrees with Appellants' arguments above, and submits that Appellants appear to read an additional limitation into the claim, in particular, a limitation that would require that the alias table elements do not change values subsequent to a mispredicted branch. The Examiner has misinterpreted Appellants' arguments above. Appellants argued that the alias table elements contain stored values that may be used to perform replacements in the event of a misprediction. In other words, Appellants have argued that the alias table elements did not change prior to the mispredicted branch. As claim 6 depends from claim 4, it requires that the dispatch unit maintains multiple alias table elements at a same time. Claim 6 then refers to "the one or more register names stored within a respective alias table element," (i.e. an alias table element associated with the microcode routine that experienced a mis-predicted branch) according to which the dispatch unit is configured to perform replacements within one or more microcode operations included within the microcode subroutine. For the stored register names to be used in performing the replacements, they must not have changed between the time they were stored and the time they are used in performing the replacements. **The citation is Carbine clearly teaches away from using stored register names, as required by claims 4 and 6, as it explicitly describes having to re-fetch and re-decode them after a mis-prediction.**

For at least the reasons above, the rejection of claim 6 is not supported by the cited art and removal thereof is respectfully requested. Claims 22 and 32 include limitations similar to claim 6, and so the arguments presented above apply with equal force to these claims, as well.

Fourth ground of rejection:

Claims 7, 8, 23, 24, 35 and 36 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran and Carbine and further in view of Rotenberg, et al. ("Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching") (hereinafter

"Rotenberg"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 7, 23, and 35:

Regarding claim 7, contrary to the Examiner's assertion, Tran in view of Carbine and Rotenberg clearly fails to teach or suggest *comprising a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the trace cache entry includes the microcode subroutine call operation and the one or more register names for use as replacement register names.*

Appellants' arguments from the Appeal Brief filed September 25, 2006 regarding this rejection are herein incorporated by reference. As explained in Appellants' Appeal Brief, for at least the reasons presented above, the cited references do not teach the microprocessor of claim 2 wherein the dynamic instruction stream includes a microcode subroutine call operation and the one or more register names for use as replacement register names, as the Examiner suggests.

Further regarding claim 7, the Examiner admits, "Tran and Carbine have not explicitly taught a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the trace cache entry includes instructions from the dynamic instruction stream". The Examiner submits that Rotenberg has taught a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry and that a trace stored in the trace cache entry includes instructions from the dynamic stream. Appellants note that the Examiner has misquoted Appellants' claim. Claim 7 does not recite that a trace cache entry includes "instructions from the dynamic stream," but a trace cache entry that includes the microcode subroutine call operation and the one or more register names for use as replacement register names. There is nothing in Rotenberg or the other cited references that teaches a trace entry including these specific elements. Therefore Tran in view of Carbine and Rotenberg does not teach or suggest all the limitations of claim 7.

The Examiner submits that it would have been obvious to a person of ordinary skill in the art to modify the system of Tran and Carbine to include a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the trace cache entry includes instructions from the dynamic instruction stream as taught by Rotenberg because the trace cache would improve the performance of the microprocessor (Rotenberg, Abstract.) Appellants assert, however, that including the trace cache of Rotenberg in the system of Tran and Carbine would not teach or suggest all the limitations of claim 7, since Rotenberg's trace cache entries do not store the elements recited in claim 7.

In his Answer, the Examiner disagrees with Appellants' arguments above, and notes that he relied on Tran and Carbine to teach a trace cache entry that includes the microcode subroutine call operation and the one or more register names for use as replacement register names, and Rotenberg to teach "a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the trace cache entry includes instructions from the dynamic instruction stream." The Examiner submits that the dynamic instruction stream, as taught by Tran in view of Carbine, includes the microcode subroutine call operation and the one or more register names for use as replacement names. However, as discussed above regarding claim 2, from which claim 7 depends, Tran in view of Carbine does not teach these limitations of the dynamic instruction stream. Furthermore, Appellants again assert that adding the trace cache of Rotenberg to the system of Tran and Carbine would not teach the specific limitations of a trace cache recited in claim 7, since Rotenberg teaches only that instructions fetched from memory, not micro-operations (i.e., the operations fetched from a microcode ROM that make up a microcode subroutine) are stored in a trace cache. In other words, the trace cache of Rotenberg is not designed to store microcode traces. It would change the principle of operation of Rotenberg's trace cache to include these microcode subroutines, and the Examiner has not explained how the trace cache of Rotenberg could be incorporated into the teachings of Tran and Carbine or how that would result in Appellants' claimed invention.

For at least the reasons above, the rejection of claim 7 is not supported by the cited art and removal thereof is respectfully requested. Claims 23 and 35 include limitations similar to claim 7, and so the arguments presented above apply with equal force to these claims, as well.

Claims 8, 24, and 36:

Regarding claim 8, as explained in Appellants' Appeal Brief, Tran in view of Carbine and Rotenberg clearly fails to teach or suggest *wherein in response to receiving the trace from the trace cache, the dispatch unit is configured to allocate an alias table to store the one or more register names.* The Examiner again cites Carbine (micro-alias register; FIG. 5, component 562; and column 12, lines 15-35) as teaching this limitation, "In response to receiving the LOADUAR instruction, a micro-alias register is allocated." However, as discussed above, LOADUAR is not an instruction at all (i.e., an operation dispatch to the scheduler for execution). Instead, it is a signal that causes micro-alias register 562 to be loaded with values from one of four Cuops. Therefore, the cited references clearly do not teach or suggest this limitation.

In his Answer, the Examiner disagrees with Appellants' arguments above, and further submits that, "the LOADUAR signal is an operation, or instruction, that loads a micro-alias register (i.e., alias table element) with a register name." **Appellants again assert that no one of ordinary skill in the art would consider the LOADUAR signal to be an operation (much less an instruction) dispatched from a dispatch unit, as recited in Appellants' claims.** Furthermore, as discussed above, the combination of Tran, Carbine, and Rotenberg does not teach **a trace stored in a trace cache that includes a microcode subroutine call operation and one or more register names.** There is nothing in the references to suggest that the combination suggested by the Examiner be made or that it could even operate according to the limitations of the present invention. Furthermore, the cited references do not teach the alias table of Appellants' invention, as discussed above in reference to claim 3.

For at least the reasons above, the rejection of claim 8 is not supported by the cited art and removal thereof is respectfully requested. Claims 24 and 36 include limitations similar to claim 8, and so the arguments presented above apply with equal force to these claims, as well.

Fifth ground of rejection:

Claims 9, 10, 25 and 37 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran in view of Kling (U.S. Publication 2004/0049657). Appellants traverse this rejection for at least the following reasons.

Regarding claim 9, contrary to the Examiner's assertion, Tran in view of Kling clearly fails to teach or suggest *wherein the microcode subroutine is stored as one or more microcode traces.*

Appellants' arguments from the Appeal Brief filed September 25, 2006 regarding this rejection are herein incorporated by reference. The Examiner admitted that Tran has not explicitly taught that the dispatch unit is configured to store the microcode subroutine in one or more microcode traces, and relied on Kling to teach this limitation (microcode unit; FIG. 2, component 46; paragraph 13; and paragraph 32). However, component 46 of FIG. 2 does not illustrate a microcode unit, as the Examiner suggests, but microcode. In addition, the Examiner has mischaracterized the teachings of paragraph [0032] of Kling, which mentions the existence of a "microcode trace" but does not teach or suggest that a microcode **subroutine** is stored in one or more such traces. The only other mention of a "trace" in Kling is found in claim 25, which includes the limitation, "using a microcode trace to store information associated with accessing the register space." Appellants assert that neither of these passages teaches or suggests the limitations of Appellants' claim 9.

In his Answer, the Examiner disagrees with Appellants' arguments above, and notes that "Clearly, microcode is a unit of processor 12 of Kling." This is completely irrelevant, as it teaches nothing about the limitations of this claim. The Examiner further submits that "a subroutine is merely a sequence of instruction" and that since Kling teaches a microcode trace, it must inherently include at least one microcode subroutine. This is incorrect. First, "subroutine" is a term of art that does not merely refer to any sequence of instructions, as would be understood by one of ordinary skill in the art. In addition, Appellants assert that it is not inherent that a microcode trace includes any microcode subroutines, since the use of microcode to implement various operations in a processor does not require that the micro-operations be organized as microcode subroutines. Instead, microcode execution flow is typically controlled using a microcode sequencer to determine the next microcode operation to be executed, not a call and return mechanism typical of a subroutine. Therefore, a microcode trace need not contain any microcode subroutines, but may merely sequences of instructions in the order that they were executed (i.e., a dynamic stream of individual micro-operations). Appellants' claim 9, as it depends from claim 1, requires *a microcode subroutine stored as one or more microcode traces* and *associated with a microcoded instruction received by a dispatch unit*, and *a microcode subroutine call operation that includes a tag identifying the microcode subroutine* being *dispatched to a scheduler*. **Appellants assert Kling clearly does not overcome the deficiencies of Tran in teaching or suggesting these specific limitations regarding a microcode subroutine by merely mentioning the existence of a "microcode trace."**

For at least the reasons above, the rejection of claim 9 is not supported by the cited art and removal thereof is respectfully requested. Claims 25 and 37 include limitations similar to claim 9, and so the arguments presented above apply with equal force to these claims, as well.

Sixth ground of rejection:

Claims 11-14, 26 and 38 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran and Kling and further in view of Harris (U.S. Patent 6,260,138). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 11, 26, and 38:

Regarding claim 11, contrary to the Examiner's assertion, Tran in view of Kling and Harris clearly fails to teach or suggest *wherein each microcode operation stored in the one or more microcode traces includes an associated liveness indication.*

Appellants' arguments from the Appeal Brief filed September 25, 2006 regarding this rejection are herein incorporated by reference. The Examiner admitted that Tran and Kling have not explicitly taught that each operation includes an associated liveness indication and relies on Harris to teach this limitation. However, as explained in Appellants' Appeal Brief, the priority tag of Harris is not an indication of liveness, as this term would be understood by one of ordinary skill in the art or as used in Appellants' specification. Instead, the priority tags of Harris are assigned individually to each instruction and are used to determine an order in which to execute instructions on different branch paths. They may be dependent on the position of each instruction with respect to a predicted path (see, e.g., FIG. 7 of Harris). However, there is nothing in Harris that describes storing such an indication along with microcode operations in a microcode trace, as recited in Appellants' claims. Instead, Harris describes that the priority tags may be stored in an instruction cache (which is not a trace cache) or in a separate buffer, as in the Examiner's citation in Harris. Harris does not describe a trace cache at all, much less a trace cache storing microcode subroutines. Furthermore, the Examiner has admitted that Tran has not explicitly taught that the dispatch unit is configured to store the microcode subroutine in one or more microcode traces (as discussed above regarding claim 9) and Appellants have shown that Kling does not

overcome the deficiencies of Tran in teaching this limitation. Therefore, it is not clear how, or even if, the priority tags of Harris could be applied to Tran and Kling, or that the addition of priority tags would improve the performance of a trace cache including microcode subroutines (which is not found in Tran or Kling). Appellants assert that since Tran in view of Kling does not teach that microcode subroutines are stored in microcode traces, the combination of Tran, Kling, and Harris cannot teach or suggest additional limitations on such microcode traces.

In his Answer (regarding claims 11, 26, and 38, and 12, 13, 14), the Examiner disagrees with Appellants' arguments above, and submits that Appellants appear to be reading "liveness indication" too narrowly. Appellants again remind the Examiner that the claims must be interpreted in a way that is consistent with the specification, and that the liveness indication of Appellants claims is clearly described in the specification. The Examiner notes that Kling (Appellants assume the Examiner means Harris) indicates that that a priority tag (also referred to as a prediction bit) indicates whether an instruction relates to a predicted or non-predicted path following a branch instruction. The Examiner submits that the priority tag indicates whether or not the instruction belongs to the group of instructions on the predicted path or the non-predicted path (which he equates to a liveness group) and is thus a liveness indication. Appellants disagree. The Examiner's statement that the priority tag, as defined by Harris, is a "liveness indication" is completely unsupported in the cited art (which does not include the term "liveness" at all) and the Examiner has not provided any evidence that this is a reasonable alternate interpretation of the term "liveness indication". The priority tag indicates which instructions are predicted to be executed and which are not, not which groups of instructions are dependent on one or more branch outcomes, as indicated by the liveness indications of the present invention.

The Examiner also states that the system of Tran and Kling stores microcode instructions within a trace and that Harris is relied upon for teaching storing the priority along with the associated instruction. However, as discussed above, Tran and Kling do not teach storing microcode subroutines in a trace (as required by claim 9, from which

claim 11 depends). Therefore the combined references still do not teach all the limitations of claim 11.

For at least the reasons above, the rejection of claim 11 is not supported by the cited art and removal thereof is respectfully requested. Claims 26 and 38 include limitations similar to claim 11, and so the arguments presented above apply with equal force to these claims, as well.

Claims 12, 13, and 14:

As explained in Appellants' Appeal Brief, for at least the reasons presented above regarding claim 11, Tran in view of Kling and Harris also clearly fails to teach or suggest the limitations of claims 12, 13, and 14, which recite additional limitations on the use of liveness indications stored with each microcode operation in one or more microcode traces. Since it has been shown (above) that the cited references do not teach liveness indications stored with each microcode operation in one or more microcode traces, they clearly cannot, and do not, teach or suggest the additional limitations of claims 12, 13, and 14.

Appellants note that the Examiner does not specifically address the rejection of these claims in his Answer, but includes them in his remarks regarding claims 11, 26, and 38, which have been addressed above.

Appellants again assert that the rejection of claims 12, 13, and 14 is not supported by the cited art and removal thereof is respectfully requested.

CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-41 is erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge any fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5500-81600/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: May 29, 2007